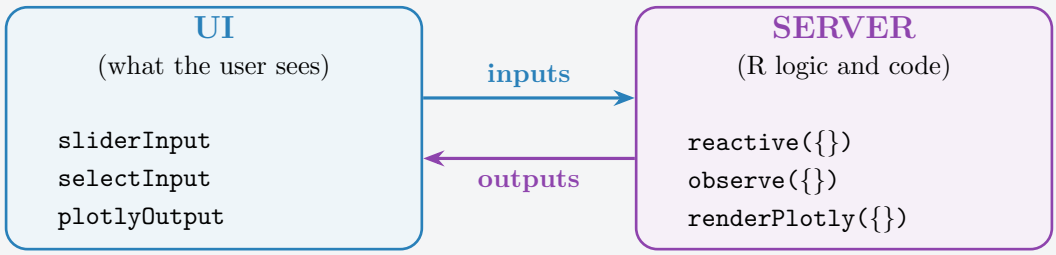
What is a Shiny App?

A Shiny app is an interactive web application built entirely in R. You do not need to know web development to build one. If you learn the right vocabulary, AI tools like Claude or ChatGPT can build one for you in minutes.

How It Works: UI + Server



- The **UI** defines the layout and input widgets (sliders, dropdowns, buttons). This is what users interact with.
- The **Server** contains R code that runs computations and produces outputs (plots, tables, text).
- **Reactivity** means the app automatically updates outputs when a user changes an input. You do not write refresh logic.
- Inputs flow from the UI to the server. Outputs flow from the server back to the UI.
- A single-file app puts both the UI and server in one file called `app.R`. Run it in RStudio with one click.

Creating a Shared Understanding

1. Input Widgets

R Function	What the User Sees
`sliderInput`	A draggable slider for picking a number in a range.
`selectInput`	A dropdown list. User picks one option from a menu.
`radioButtons`	A set of circles. User picks exactly one option. All options visible at once.
`checkboxInput`	A single on/off toggle (true or false).
`checkboxGroupInput`	Multiple checkboxes. User can select any combination.
`numericInput`	A small box for typing a number.
`textInput`	A small box for typing text.
`dateInput`	A calendar picker for selecting a date.
`fileInput`	A file upload button.

2. Output Types

In the UI	What It Displays
`plotlyOutput`	An interactive plot. Users can hover, zoom, and pan.
`plotOutput`	A static plot (from ggplot2 or base R).
`DTOutput`	An interactive data table with search, sort, and pagination.
`textOutput`	A line of plain text.
`downloadButton`	A button that lets users download a file (CSV, PDF, etc.).

3. Layout and Containers

R Function	What It Does
`page_sidebar`	Full page with a sidebar on the left and a main panel on the right. The most common starting layout.
`page_navbar`	Full page with a navigation bar across the top. Each tab is a separate view.
`sidebar`	The left panel that holds input widgets.
`card`	A bordered rectangle that holds content (plots, text, tables). The main building block.
`nav_panel`	A single tab inside a navbar page. Each tab gets a title.
`layout_column_wrap`	Arranges items in a responsive grid (side by side on wide screens, stacked on narrow ones).

4. Styling and CSS

Term	What It Controls
CSS	The language that sets visual style: colors, fonts, spacing, layout. Shiny apps use CSS behind the scenes.
`padding`	Space *inside* an element's border. Makes content feel less cramped.
`margin`	Space *outside* an element's border. Pushes elements apart from each other.
`background-color`	The fill color behind content. Set with hex codes like `#2D2D2D`.
`font-size`	How big the text is. Use `em` (relative) or `px` (absolute).
`bs_theme()`	R function that sets the overall look of the app (colors, fonts, dark mode) in one place.

5. Reactivity (How the App Stays Alive)

Term	What It Does
`input$x`	Reads the current value of a UI widget inside the server. The name `x` matches the widget's ID.
`output$x`	A slot where the server sends rendered content back to the UI. Matched to the UI output by ID.
`reactive({})`	A block of R code that automatically re-runs when any input it uses changes. Returns a value.
`observe({})`	A block that re-runs when inputs change but does not return a value. Used for side effects (updating widgets, logging).
`render*()`	Functions that produce output for the UI. Each output type has one: `renderPlotly`, `renderDT`, `renderText`.

Working with AI to Build Your App

You do not need to memorize the terms on the previous page. You need to *recognize* them so you can use the right words when talking to AI. Keep this handout nearby as a reference.

The Difference Vocabulary Makes

Without the vocabulary

"Make me a Shiny app with a graph that changes."

"Add a dropdown that hides stuff."

"Make it look better."

With the vocabulary

"Create a `page_sidebar` app with a `sliderInput` for sample size and a `plotlyOutput` histogram."

"Add a `selectInput` dropdown. When the user selects 'custom', show a `numericInput` below it."

"Apply a dark `bs_theme`. Add 1em padding to the card body."

How to Get Started

1. **Describe what you want in plain English first.** "I want a tool where someone picks a dataset from a dropdown, sees a scatterplot, and can download the filtered data as a CSV."
2. **Then layer in the vocabulary.** "Use a `page_sidebar` layout. The dropdown should be a `selectInput` in the sidebar. The scatterplot should be a `plotlyOutput`. Add a `downloadButton` for the CSV."
3. **Ask for a single-file `app.R`.** Everything in one file, runnable with one click in RStudio.
4. **Run it immediately.** Do not wait until it is perfect. Run it, see what is wrong, and tell the AI what to fix.
5. **Fix one thing at a time.** "The plot title is too small. Set `font-size` to 1.2em." "The sidebar is too wide. Set width to 250."

Practical Tips

- **Start with your data.** Bring a CSV you already care about. Building around real data is more useful than starting from scratch.
- **Browse the Shiny Gallery.** `shiny.posit.co/r/gallery` has dozens of apps with source code. Find one close to what you want and adapt it.
- **Screenshot what you like.** See a dashboard you want? Screenshot it. "Make something like this" is a valid AI prompt.
- **Run early, run often.** Get one widget and one output working first. Add features one at a time.
- **Read the error messages.** Usually a missing comma or uninstalled package. The error tells you where to look.
- **Search before you build.** Stack Overflow and Posit Community have answers to most Shiny questions already.
- **Deploy for free.** `posit.cloud` hosts up to 5 apps free. Your app gets a public URL you can share.
- **Iterate with AI.** Give it your code and a screenshot. Fix one thing at a time. You can always say "undo that."

Packages Worth Knowing

- `shiny` – the core framework
- `bslib` – modern layout and theming
- `plotly` – interactive plots
- `ggplot2` – standard R plotting
- `DT` – interactive data tables
- `shinyWidgets` – extra input widgets

CSS Phrases for Your Prompts

- Dark background → "`background-color: #2D2D2D`"
- More space inside → "`padding: 1em`"
- More space between → "`margin: 1em`"
- Bigger text → "`font-size: 1.2em`"
- Rounded corners → "`border-radius: 8px`"
- Change text color → "`color: #C5A55A`"
- Don't know hex codes? Just say "dark charcoal with gold accents."

Demo app: `demo_app.R` (included) **Live examples:** `shiny.austinsemmel.com` **Downloads:** `austinsemmel.com/code`

Getting Set Up

Everything you need is free. This page walks you from zero to a running Shiny app.

Step 1: Install R

R is the programming language that powers Shiny apps.

1. Go to `cran.r-project.org`
2. Click the download link for your OS.
3. Run the installer. Accept all defaults.

Step 2: Install RStudio

RStudio is where you write and run R code. It is free.

1. Go to `posit.co/download/rstudio-desktop/`
2. Download the free version for your OS.
3. Run the installer. Accept all defaults.

Step 3: Install the Packages

Packages add capabilities to R. Install them once. Open RStudio, paste this into the **Console** (bottom-left panel), and press Enter:

```
install.packages(c("shiny", "bslib",
  "plotly", "ggplot2", "DT"))
```

Red text will scroll by. That is normal. Wait for it to finish.

Step 4: Run Your First App

1. In RStudio: **File** → **New File** → **Shiny Web App**. Give it a name and click **Create**.
2. RStudio generates a working app (the Old Faithful geyser demo).
3. Click the **Run App** button in the top-right corner of the code editor (green play button).
4. The app opens in a new window. Move the slider and watch the plot update.
5. To stop the app, close the window or click the red stop sign in the Console.

Step 5: Build Your Own

Open a new file (**File** → **New File** → **R Script**). Save it as `app.R`. Open an AI tool (Claude, ChatGPT, etc.) and describe what you want. Use the vocabulary from page 2 to be specific. Paste the code the AI gives you into your file and click **Run App**.

Step 6: Share It

When your app is ready, you can publish it so anyone with a browser can use it.

1. Create a free account at `posit.cloud`.
2. In RStudio, click the **Publish** button (blue icon, top-right of the app viewer).
3. Follow the prompts to connect your account. Click **Publish**. Your app gets a public URL.

URLs

R download	`https://cran.r-project.org`
RStudio download	`https://posit.co/download/rstudio-desktop/`
Free app hosting	`https://posit.cloud`
Shiny Gallery	`https://shiny.posit.co/r/gallery/`
Workshop materials	`https://austinsemmel.com/code`